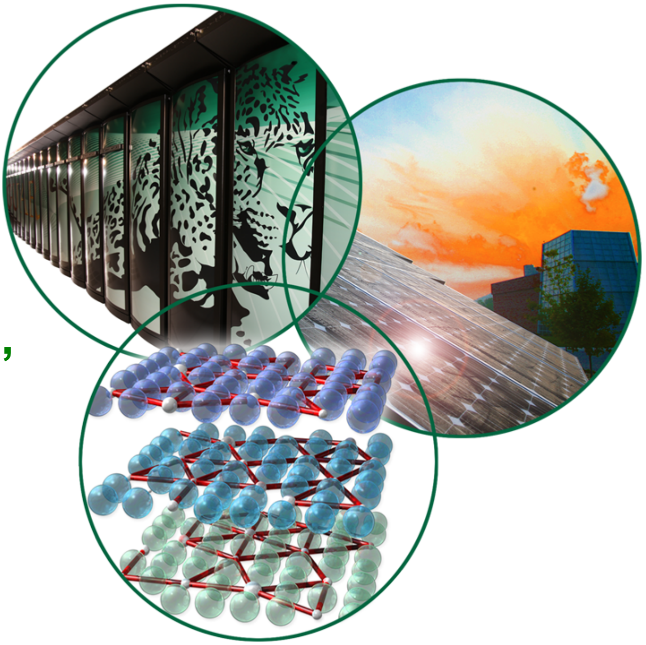
D-FACTOR: A Quantitative Performance Model of Application Slow-down in Multi-Resource Shared Systems

Presenter: Youngjae Kim

June 14th 2012



Seung-Hwan Lim[1,2], Jae-Seok Huh[1], Youngjae Kim[1], Galen M. Shipman[1], and Chita R. Das[2]

[1]Oak Ridge National Laboratory

[2]Pennsylvania State University

Oak Ridge National Laboratory

MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

U.S. DEPARTMENT OF ENERGY

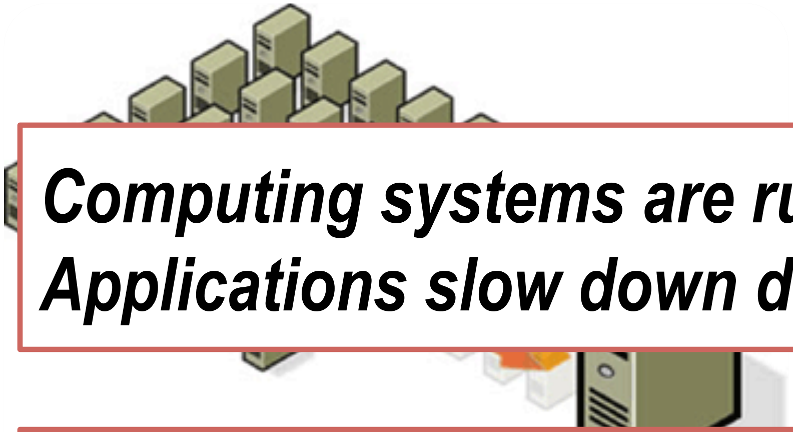
PENNSTATE

1 8 5 5

A norm in a computing system: multiple concurrent workloads

Enterprise-scale system : server consolidation

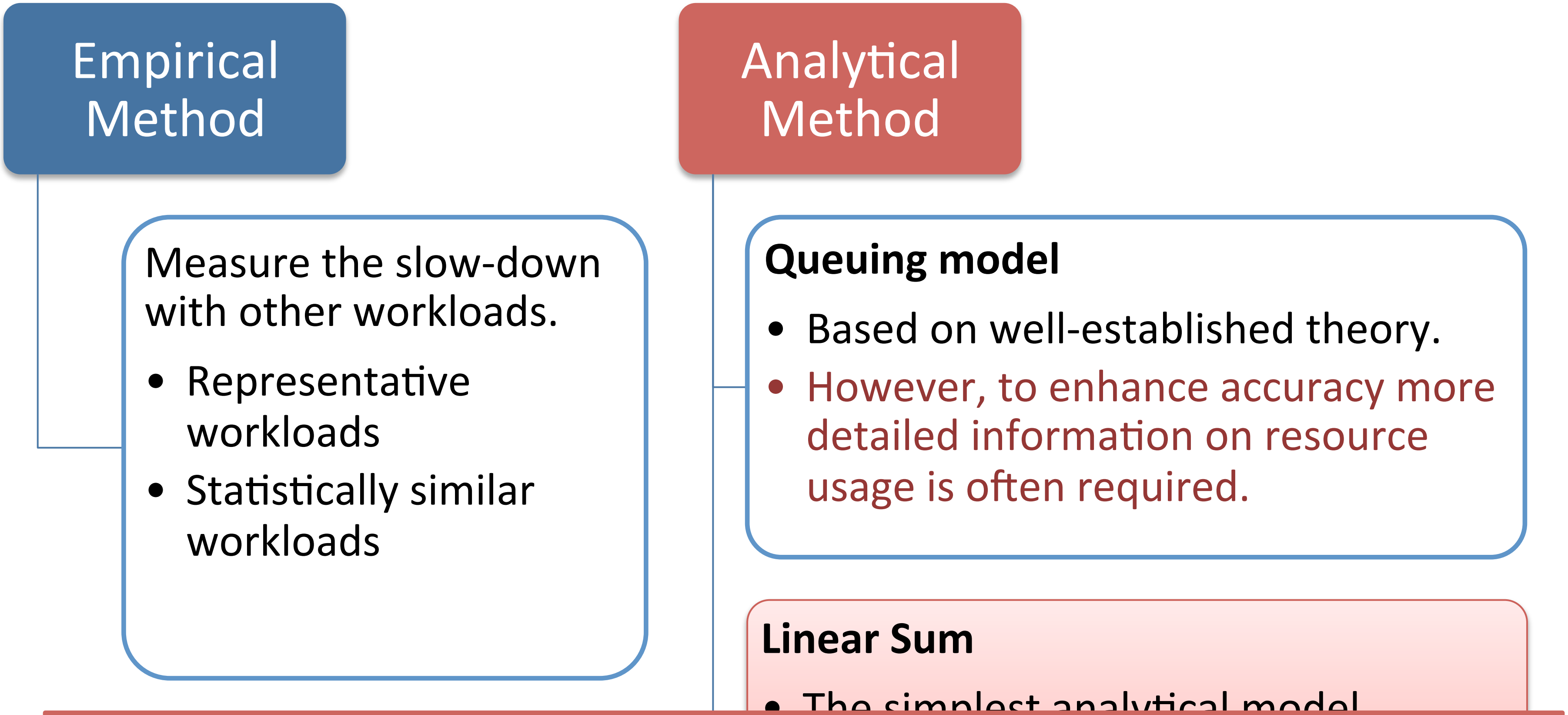
Desktop system or Smartphone : multiple programs



Computing systems are running multiple workloads. Applications slow down due to resource contentions.

How can we estimate the slow-down of multiple concurrent workloads in multi-resource systems?

Estimating the slow-down of applications due to interference.



Empirical Method

Measure the slow-down with other workloads.

- Representative workloads
- Statistically similar workloads

Analytical Method

Queuing model

- Based on well-established theory.
- However, to enhance accuracy more detailed information on resource usage is often required.

Linear Sum

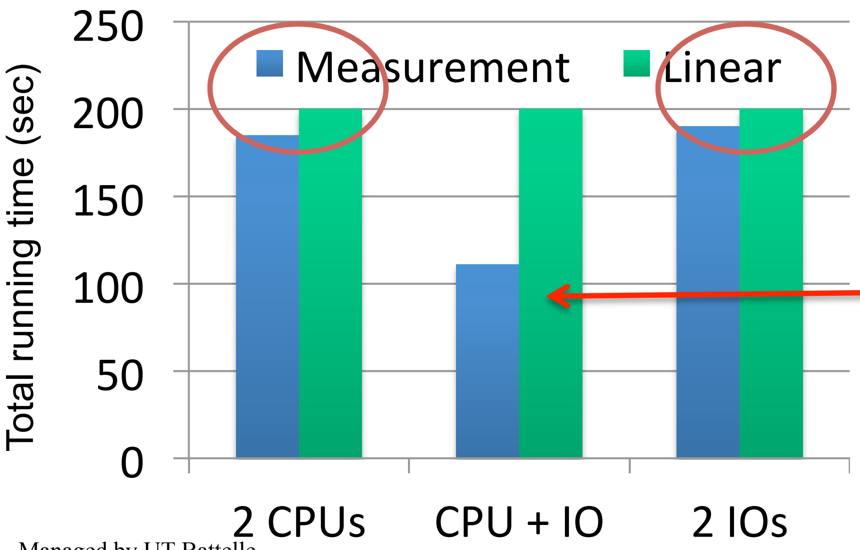
- The simplest analytical model

We extend the linear sum model to estimate the slow-down of applications due to resource contention.

The non-linear slow-down in multi-resource systems

Experiments

- **CPU workload:** **CPU job consists of arithmetic operations only**
  - **Dedicated to run on a single-core CPU**
- **I/O workload:** **Each I/O job randomly reads two 2GB of files (RAM = 4GB)**
- **Both CPU and I/O workloads take 100 sec without the presence of other workloads.**



Linear sum model fails to explain multi-resource contention.

D-Factor (*Dilation Factor*) model

Estimates the slow-down of jobs due to contention for multiple resources in a system

D-factor model extends linear sum.

Objective

We want to describe the slow-down of applications in multi-resource systems

Design Constraints

To maintain the simplicity instead of the perfection.

To easily use in existing schedulers.

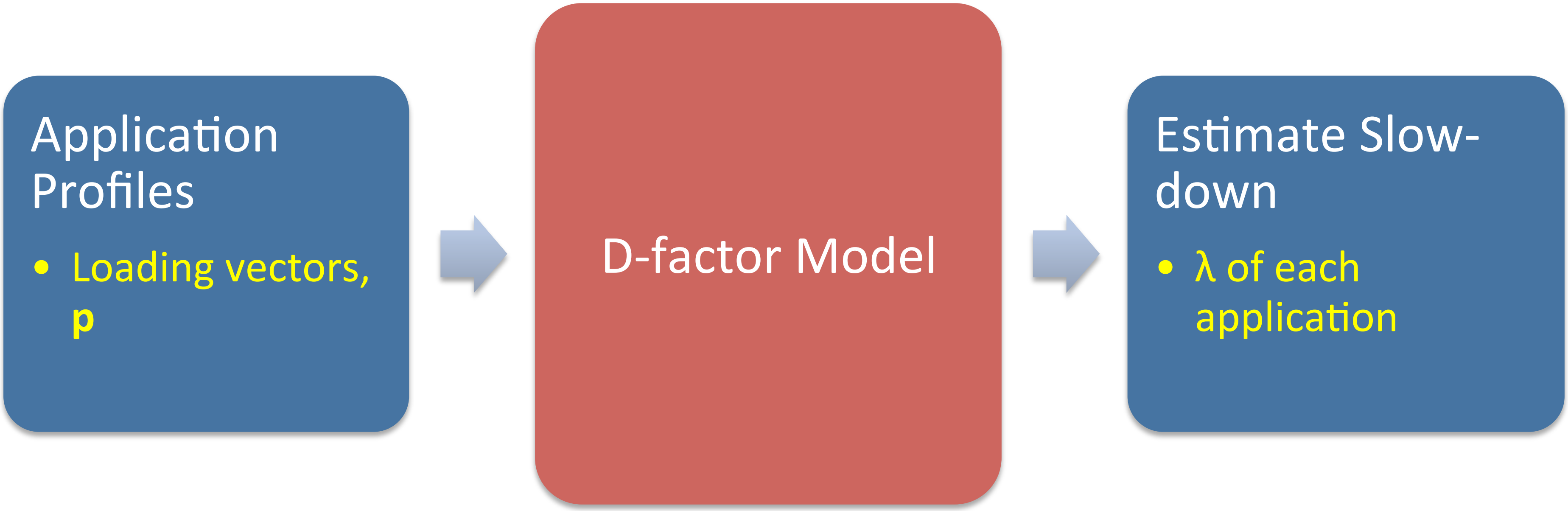
Our Approach

We extend the linear sum model. However, it has the following limitation.

The linear sum is for single-resource systems.

However, the basis of many scheduling algorithms requires to consider multi-resource system environment.

An Overview of D-factor Model Framework



Application Profiles

- Loading vectors, **p**

D-factor Model

Estimate Slow-down

- λ of each application

D-factor model explains the expected slow-down when applications are concurrently running.

λ is a quadratic function of loading vectors in the D-factor model.

Outline

Introduction

How to describe jobs and machines

- Dilation factor; job and job slices; and loading vector

How to estimate running times

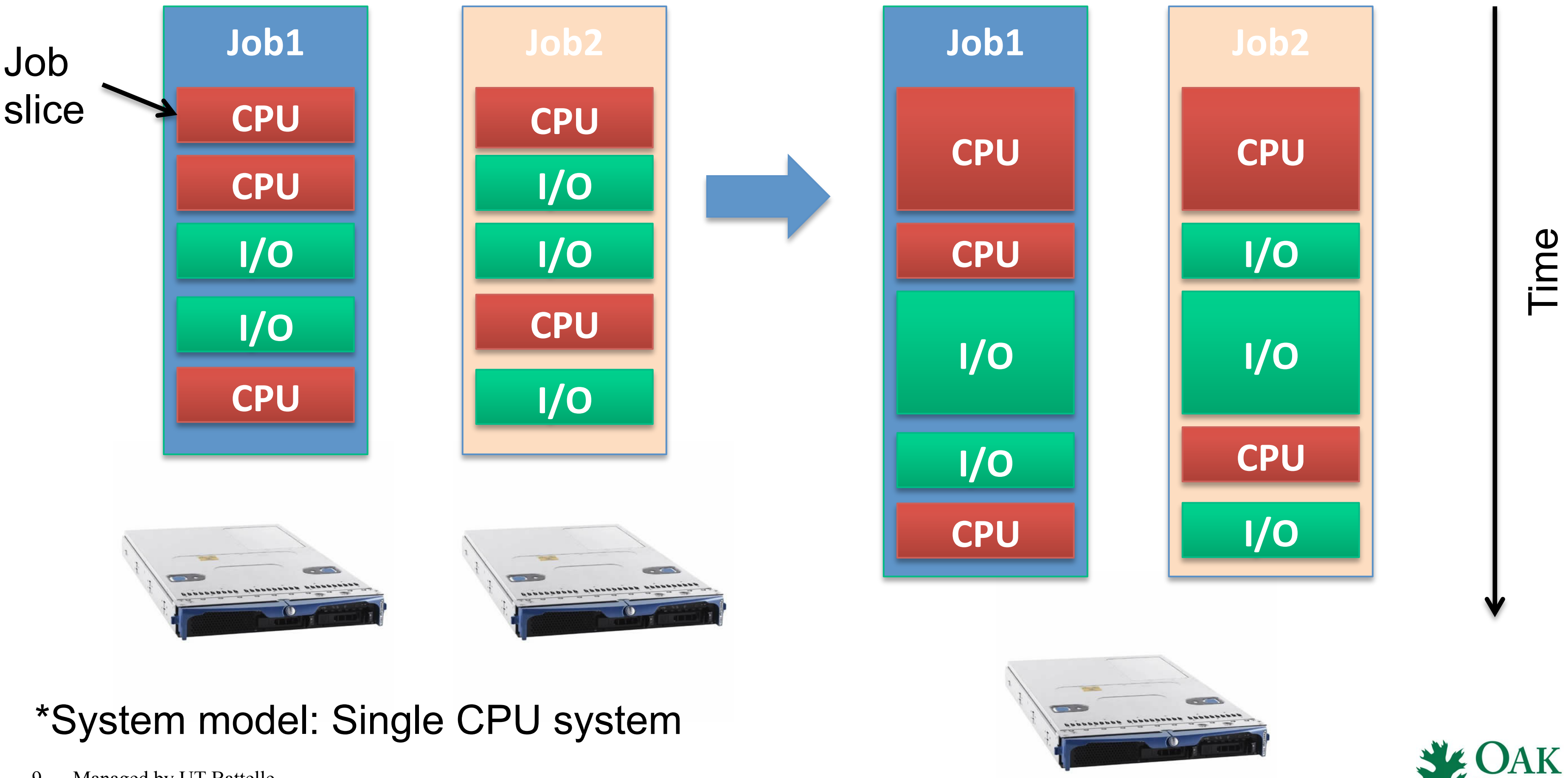
Validation results

Conclusions & Future work

Each fraction of a job will be dilated by resource contention.

Stand-alone behavior

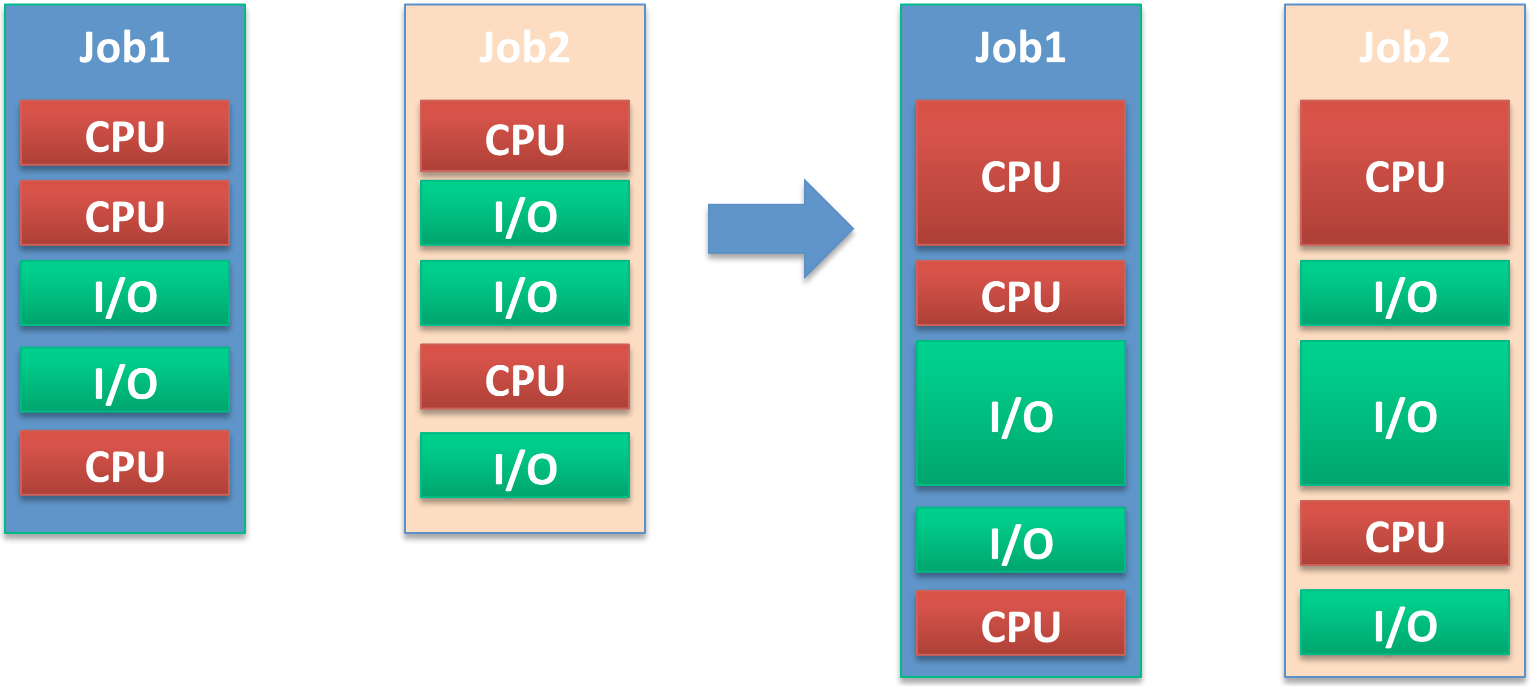
Co-located behavior



*System model: Single CPU system

Dilation Factor, λ

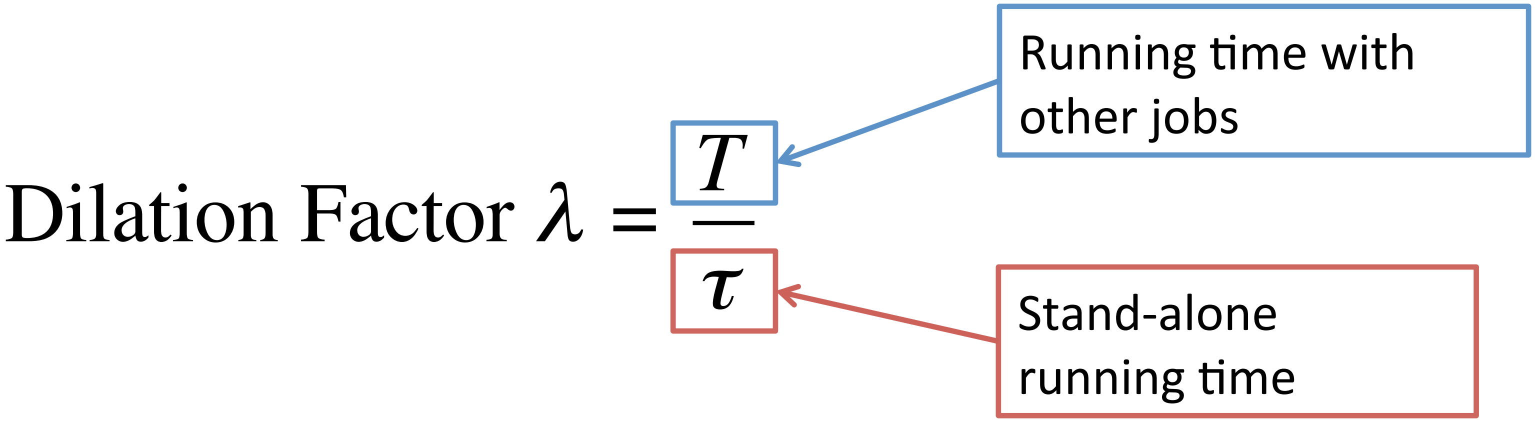
$$\lambda = \frac{\text{Running Time w/ Other Jobs}}{\text{Stand-Alone Running Time}}$$



$$\lambda_1 = \lambda_2 = 7/5 = 1.4$$

Dilation Factor
Slow-down due to resource contention

Definition 1: Dilation Factor
Dilation factor λ is the expectation of the factor of dilated completion time due to the resource contention, denoted by

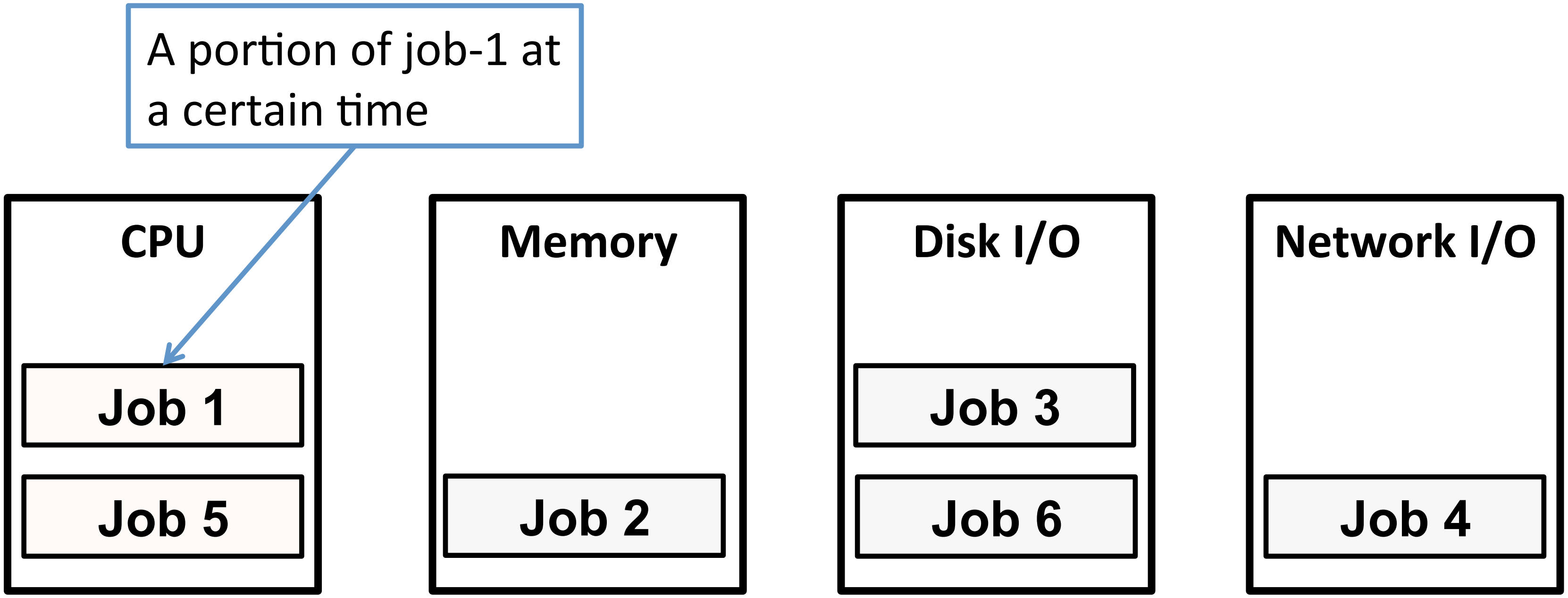


$$\text{Dilation Factor } \lambda = \frac{\boxed{T}}{\boxed{\tau}}$$

Running time with other jobs

Stand-alone running time

Machine : serves multiple jobs with shared system resources

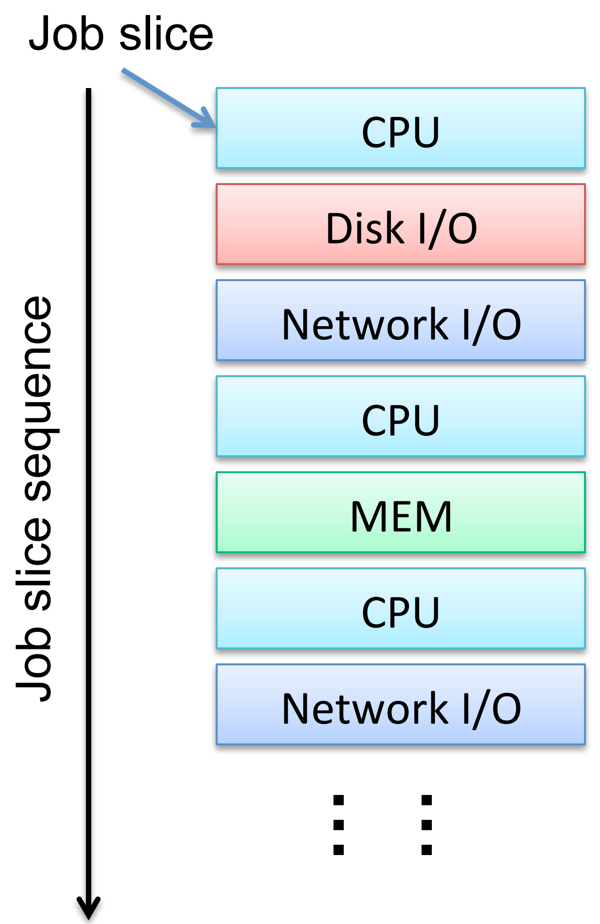


A job may contend for multiple system resources with other jobs in its overall execution.

Definition 2. Job slice and Job

Job slice : a hypothetical fraction of a job that accesses one resource

Job : a sequence of job slices

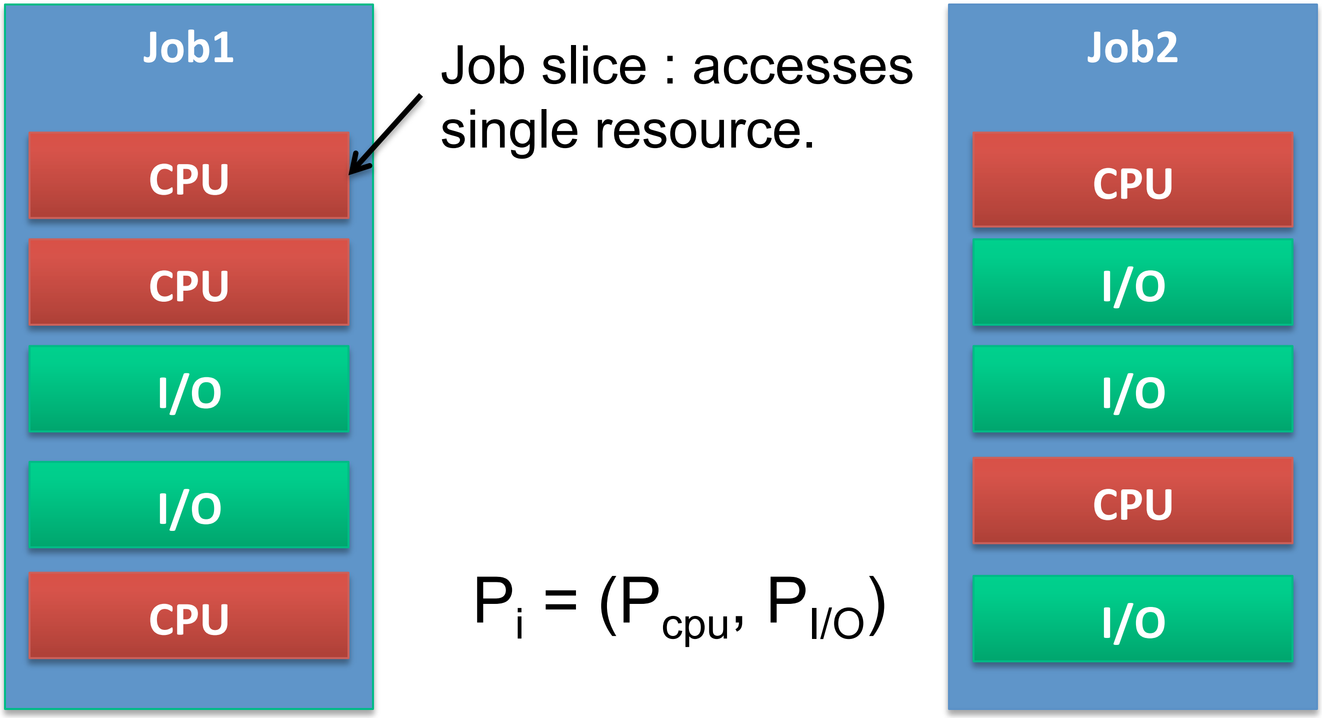


Assumptions

- **A job is a sequence of job slices.**
- **A job slice accesses only one resource for a hypothetical one-unit time.**
- **The service time of each job slice does not change by interference.**
- **No idle period between job slices.**
- **Jobs are independent to each other, i.e., different processes.**

Job : described by resource access probabilities

2 Resources (CPU and I/O) in a system



Job slice : accesses single resource.

$P_i = (P_{cpu}, P_{I/O})$

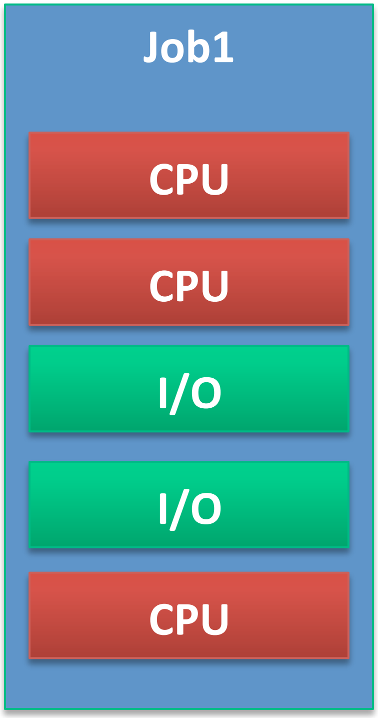
$$\mathbf{p}_1 = (0.6, 0.4)$$

Resource probability Vector P1 for Job 1

$$\mathbf{p}_2 = (0.4, 0.6)$$

Resource probability vector P2 for Job 2

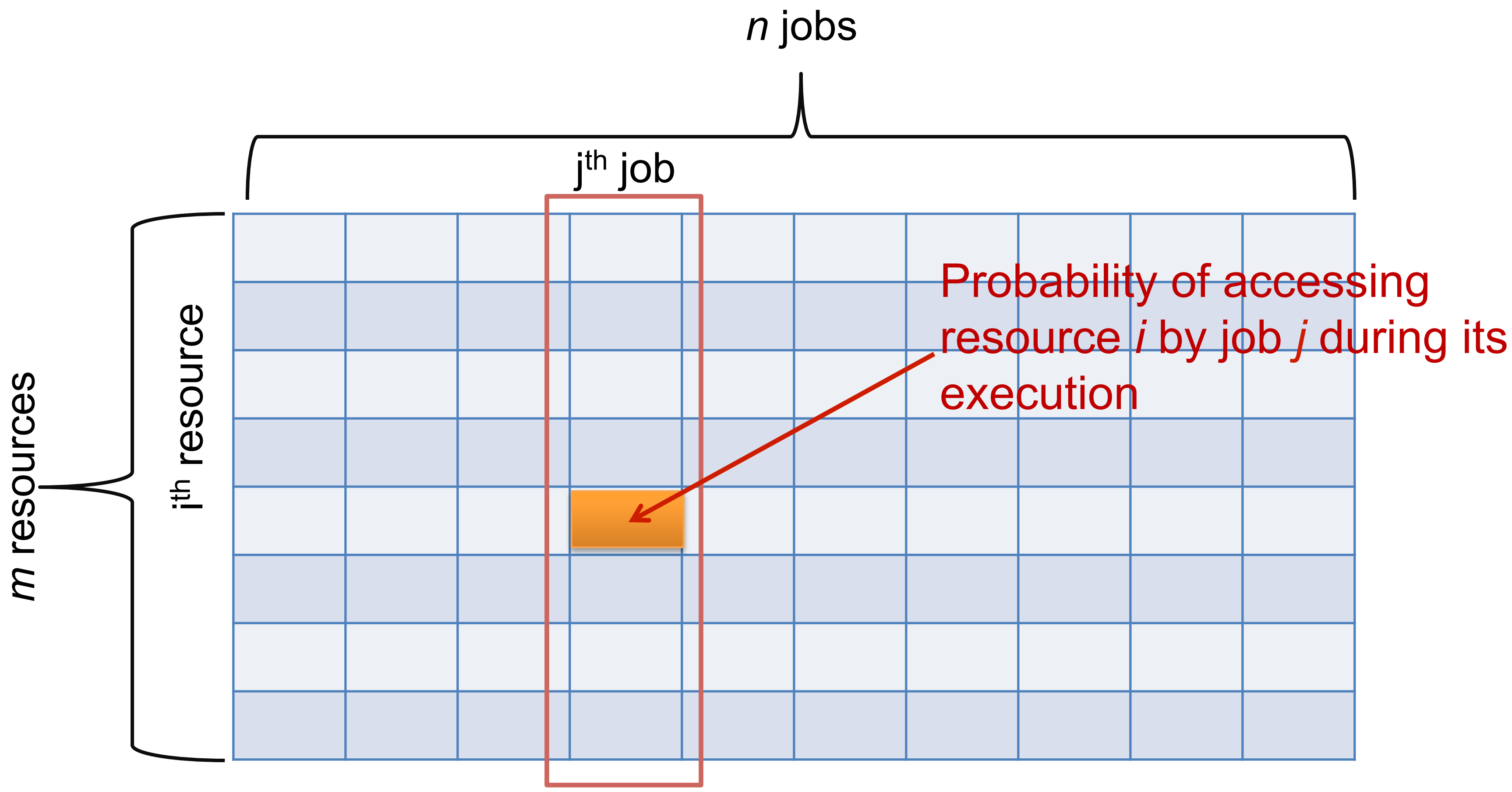
Definition 3. Loading vector :
A loading vector consists of elements that represent the portion of time in accessing each resource during execution of a job



$$\mathbf{p_1 = (0.6, 0.4)}$$

Loading vector : the statistical characterization of a job

Loading Matrix : Describes the Set of Jobs in a System



Probability of accessing resource *i* by job *j* during its execution

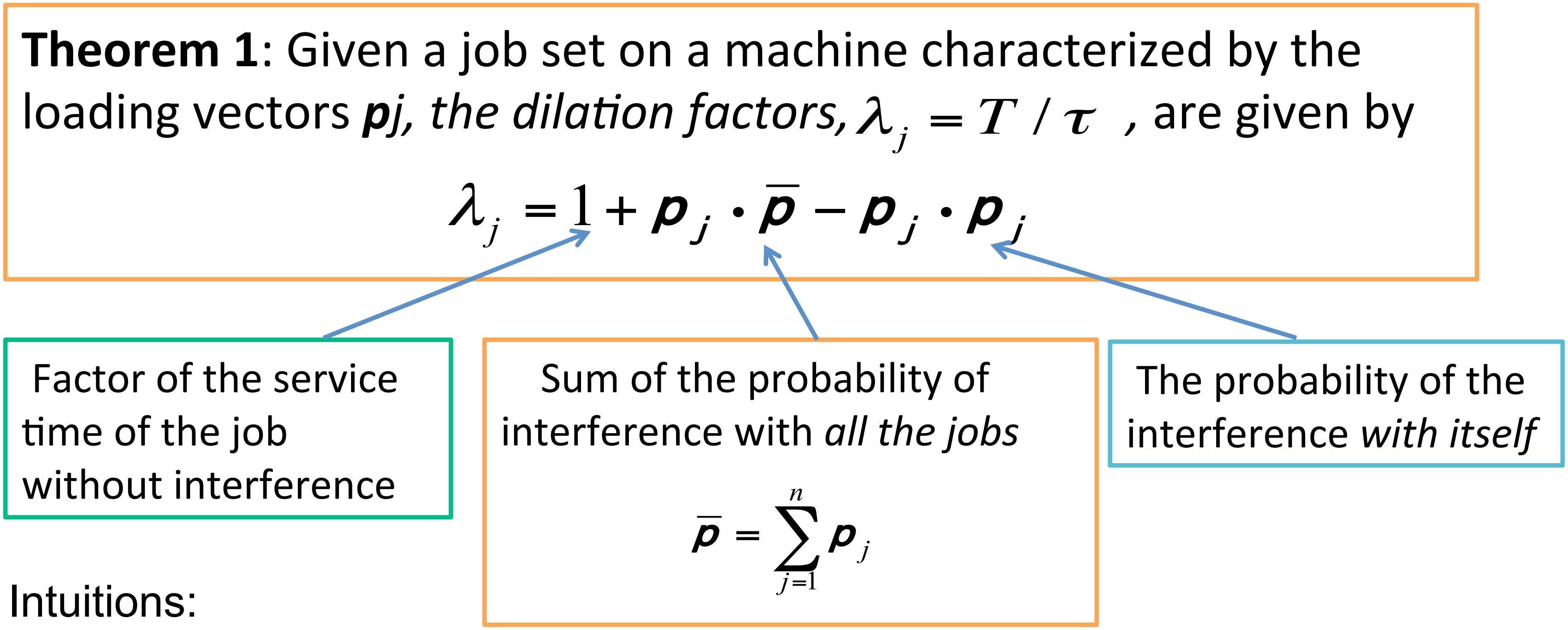
Loading vector of job *j*, $\boldsymbol{p}_j$

Total loading vector, $\bar{\boldsymbol{p}} = \sum_{j=1}^{n} \boldsymbol{p}_j$

Outline

- Introduction
- How to describe jobs and machines
- How to estimate running times
  - An example : n-jobs in 2-resource
  - By-products
    - How to obtain loading vectors of jobs
    - How to reduce to linear sum
- Validation results
- Conclusions & Future work

Dilation Factor Theorem



Theorem 1: Given a job set on a machine characterized by the loading vectors ***pj**, the dilation factors,* $\lambda_j = T / \tau$ *,* are given by

$$\lambda_j = 1 + \boldsymbol{p}_j \cdot \overline{\boldsymbol{p}} - \boldsymbol{p}_j \cdot \boldsymbol{p}_j$$

Factor of the service time of the job without interference

Sum of the probability of interference with *all the jobs*

$$\overline{\boldsymbol{p}} = \sum_{j=1}^{n} \boldsymbol{p}_j$$

The probability of the interference *with itself*

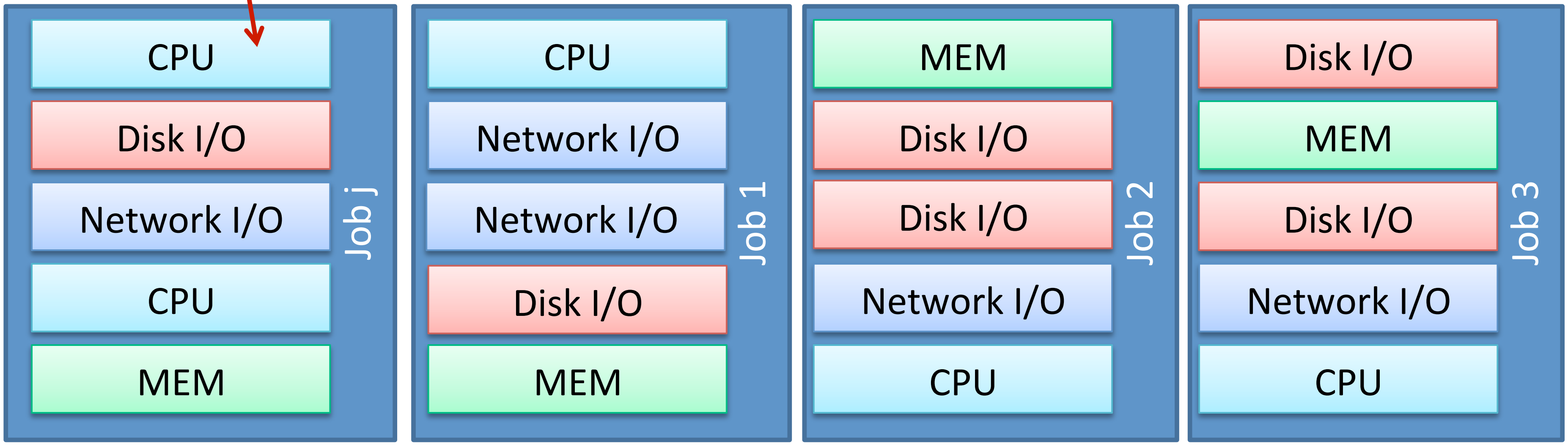
Intuitions:

Due to the resource contention, each job slice will be dilated such that from δ to δ + waiting time while other jobs are served in the resource

Theorem 1: Given a job set on a machine characterized by the loading vectors ***pj****, the dilation factors,* $\lambda_j = T / \tau$, are given by

$$\lambda_j = 1 + \boldsymbol{p}_j \cdot \bar{\boldsymbol{p}} - \boldsymbol{p}_j \cdot \boldsymbol{p}_j$$

This job slice will only wait for job1's job slice.



Job j	Job 1	Job 2	Job 3
CPU	CPU	MEM	Disk I/O
Disk I/O	Network I/O	Disk I/O	MEM
Network I/O	Network I/O	Disk I/O	Disk I/O
CPU	Disk I/O	Network I/O	Network I/O
MEM	MEM	CPU	CPU

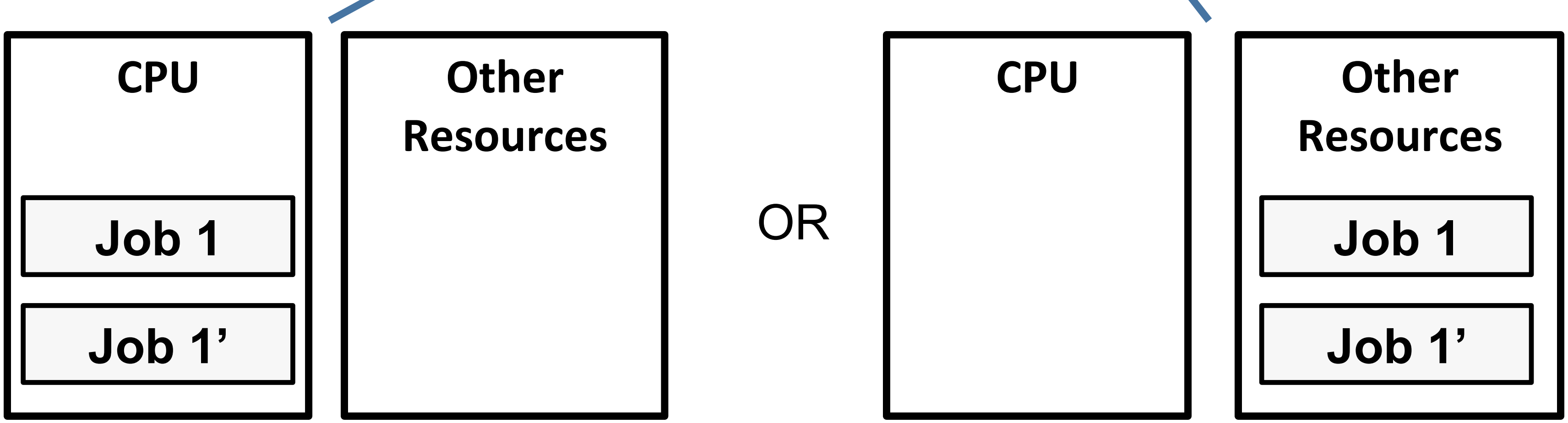
The processing time of job j's job slice dilates according to the probability of resource contention.

2-Resource, *n* Identical Jobs

Theorem 2: Assume *n* 2-resource identical jobs with the loading vector given by (p, 1-p). Then, the dilation factors are identically given by

$$\lambda = 1 + (n-1)(p^2 + (1-p)^2)$$

Intuitions: When we take non-requested resources out of consideration, the loading vector ***p*** = (*p*, *1-p*)



How to profile applications

Measure the resource usage

- Not discussed in this study.

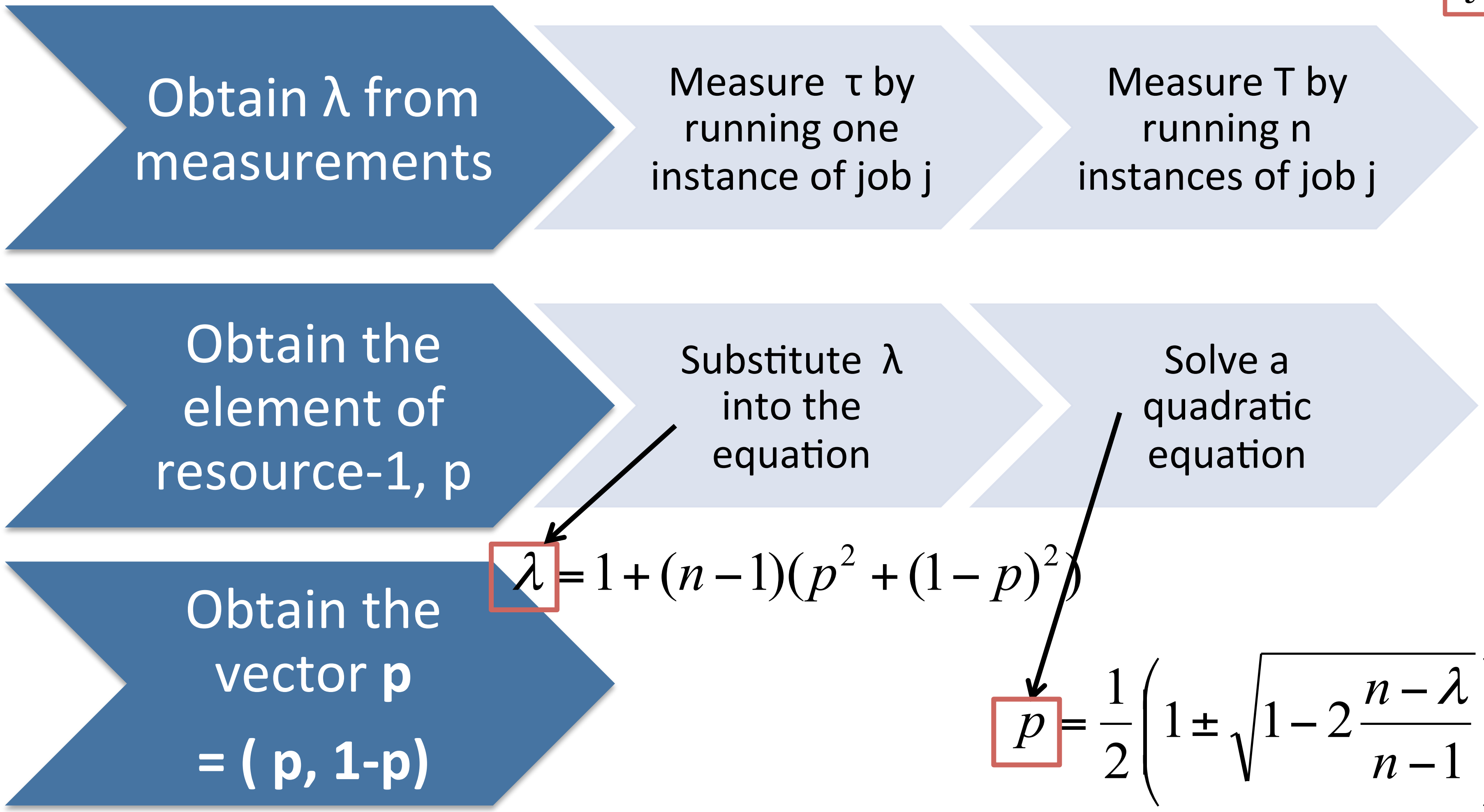
Measure the slow-down with two instances of the application.

Measure the slow-down with another well-known application.

- Included in this study.

Procedure to Obtain Loading Vector

Dilation Factor $\lambda = \frac{\boxed{T}}{\boxed{\tau}}$



Obtain λ from measurements

Measure τ by running one instance of job j

Measure T by running n instances of job j

Obtain the element of resource-1, p

Substitute λ into the equation

Solve a quadratic equation

Obtain the vector **p** **= (p, 1-p)**

$$\boxed{\lambda} = 1 + (n-1)(p^2 + (1-p)^2)$$

$$\boxed{p} = \frac{1}{2}\left(1 \pm \sqrt{1 - 2\frac{n-\lambda}{n-1}}\right)$$

1-resource jobs: linear completion time

Theorem 3: Given a job set, J, on a machine with only one resource, the total completion time of jobs , T(J) is given by the linear sum of individual job completion times, that is,

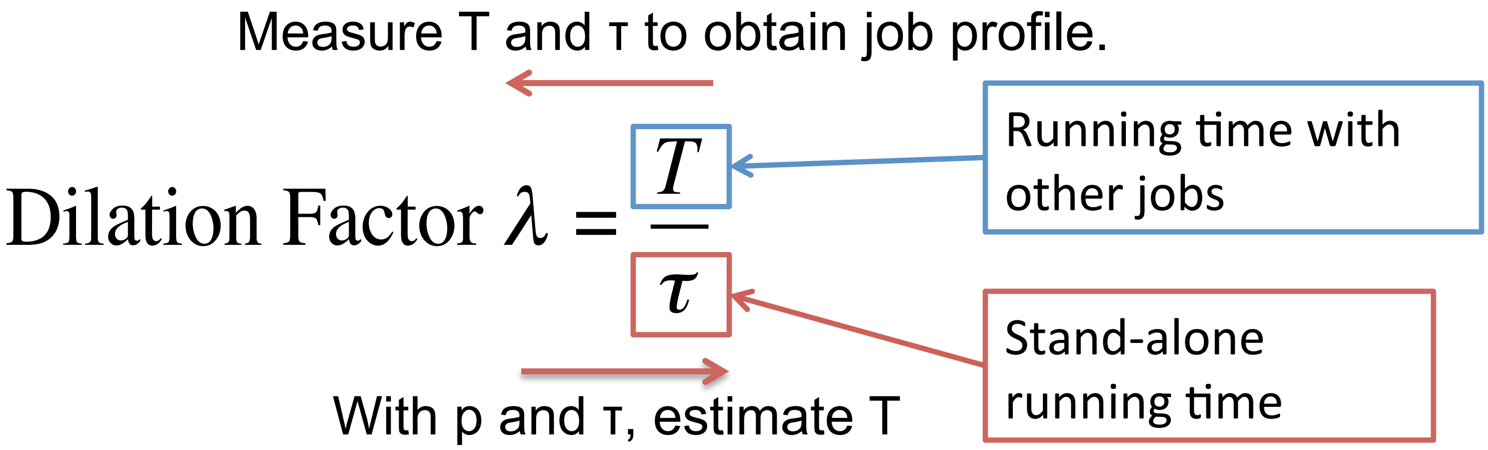
$$T(J) = \sum_{j \in J} T(j)$$

Linear sum is a special case of the dilation factor theorem

Dilation Factor is the slow-down.

We explain the relationship between the job profile, loading vector p and dilation factor λ.

We demonstrate that we can profile jobs and estimate slow-down of jobs before locating them.



Measure T and τ to obtain job profile.

$$\text{Dilation Factor } \lambda = \frac{\boxed{T}}{\boxed{\tau}}$$

Running time with other jobs

Stand-alone running time

With p and τ, estimate T

Outline

Introduction

How to describe jobs and machines

How to estimate running times

Validation results

- Workloads
- System specification
- Synthetic workloads
- Application Benchmark :FileBench (fileserver/varmail)
- MapReduce : identical jobs/non-identical jobs

Conclusions and Future work

Validating D-Factor Model

D-factor model can provide

1. More accurate estimation of the completion times of co-hosted jobs than the linear sum model
2. More efficient utilization of the system resource
3. Better predictable performance with existing scheduling algorithms than with the linear sum model

Experimental Setup

Experimented with synthetic and realistic workloads

Experimented on native Linux and Xen-based VM environment

Ran 40 times for each case and presented average values

Description of Workloads

🟥 Virtualized 🟦 Native 🟧 Both

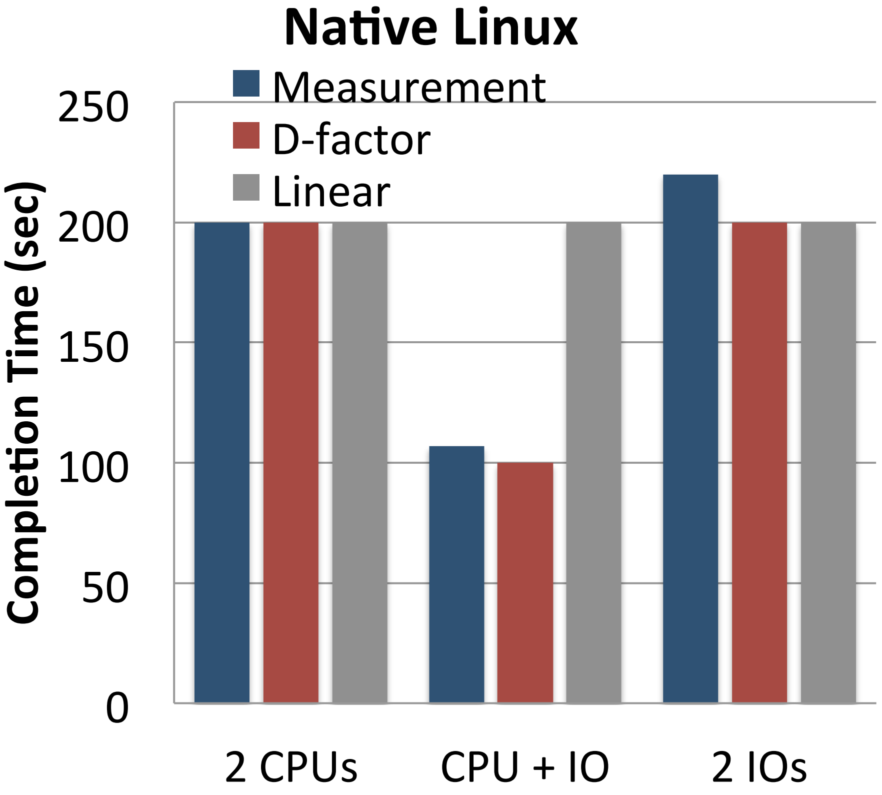
	Workload	CPU	Mem	I/O
Synthetic	CPU	**High**	**Low**	**Low**
	I/O	**Low**	**Low**	**High**
FileBench	Fileserver	**Low**	**High**	**High**
	Mailserver	**High**	**Medium**	**Medium**
MapReduce	Sort (1GB)	**High**	**High**	**Low**
	Grep	**Medium**	**High**	**Medium**
	PiEstimator	**Medium**	**High**	**Medium**

System specification

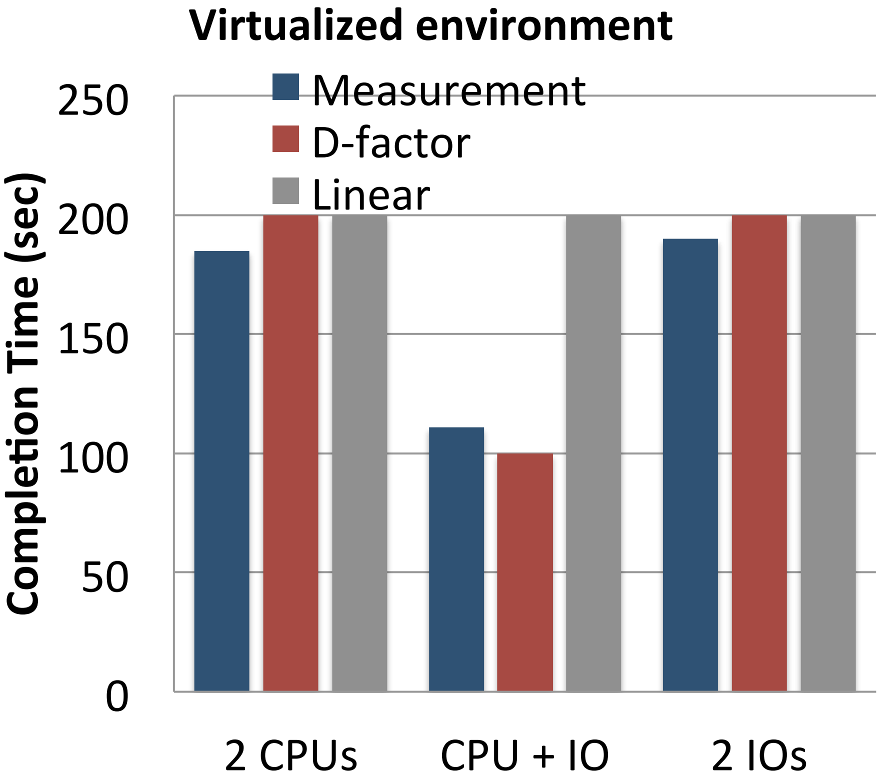
Parameters	Values
CPU	Two single-core 64bit AMD 2.4GHz
RAM	4GB
Shared Storage	NFS, disk images for Xen
Local Storage	Ultra320 SCSI
Network	1Gbps Ethernet to NFS, 10Gbps Infiniband between nodes
vCPU (Dom0)	Runs on both CPUs
vCPU (VMs)	Runs on one CPU
RAM/VM	256MB
I/O (VM)	TAP:AIO (bypasses buffer cache of Dom-0)
Kernel	Linux 2.6.18
Hypervisor	Xen 3.4.2

Validation : Synthetic workloads

error < 7%



error < 10%

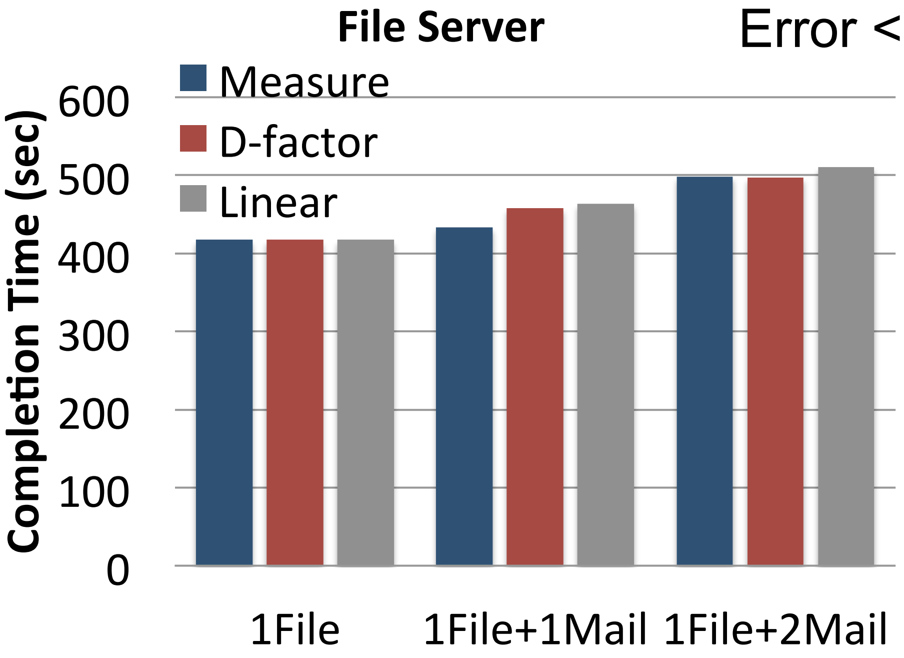


CPU : consists of arithmetic operations only

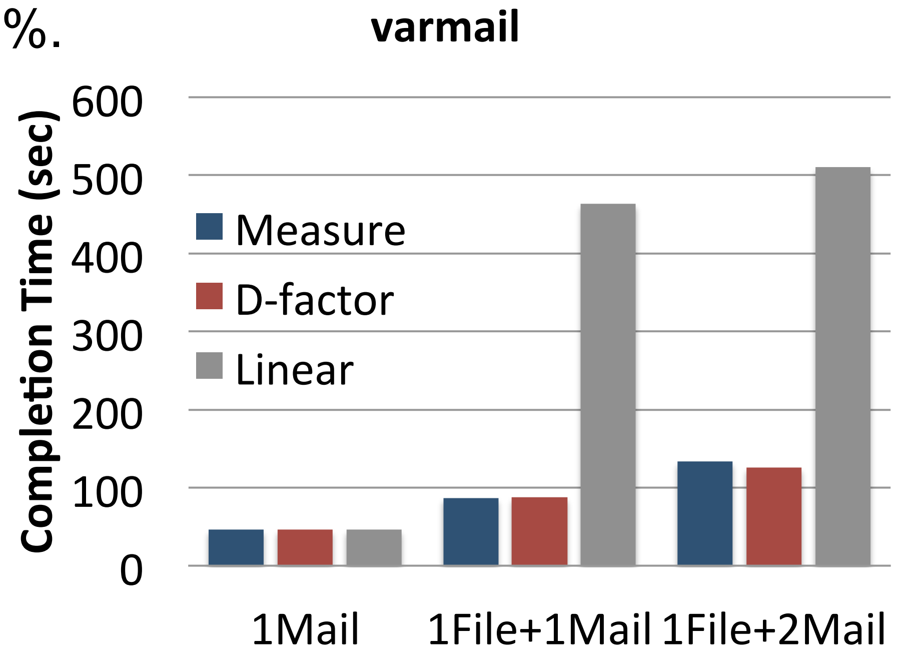
IO : reads two 2GB files

Validation : FileBench workloads

Each workload hosted in separate virtual machines.



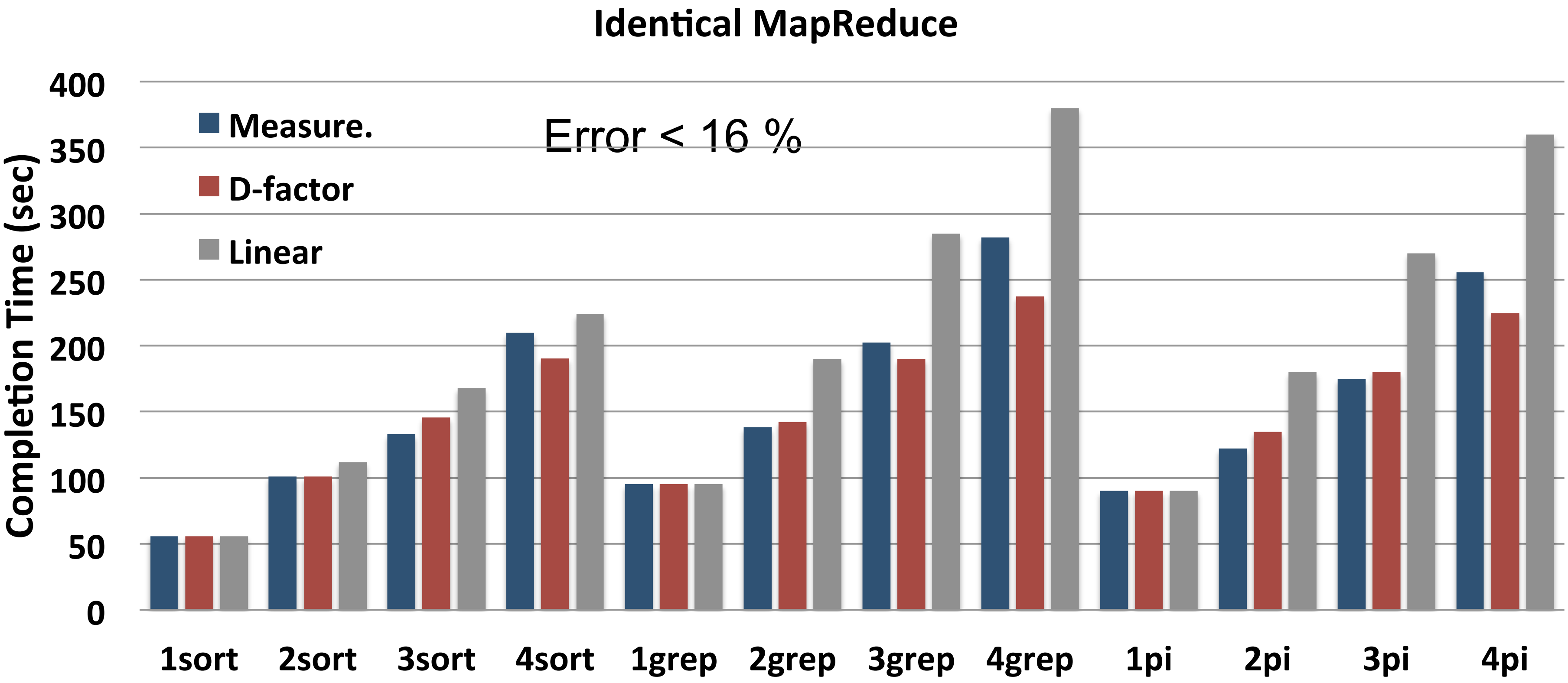
Error < 6%.



D-factor can estimate the slow-down of each job while Linear sum can't. Recall that D-factor is an extension of Linear sum.

Validation: MapReduce workloads

A 17 node Hadoop cluster results (1 master, 16 slaves)

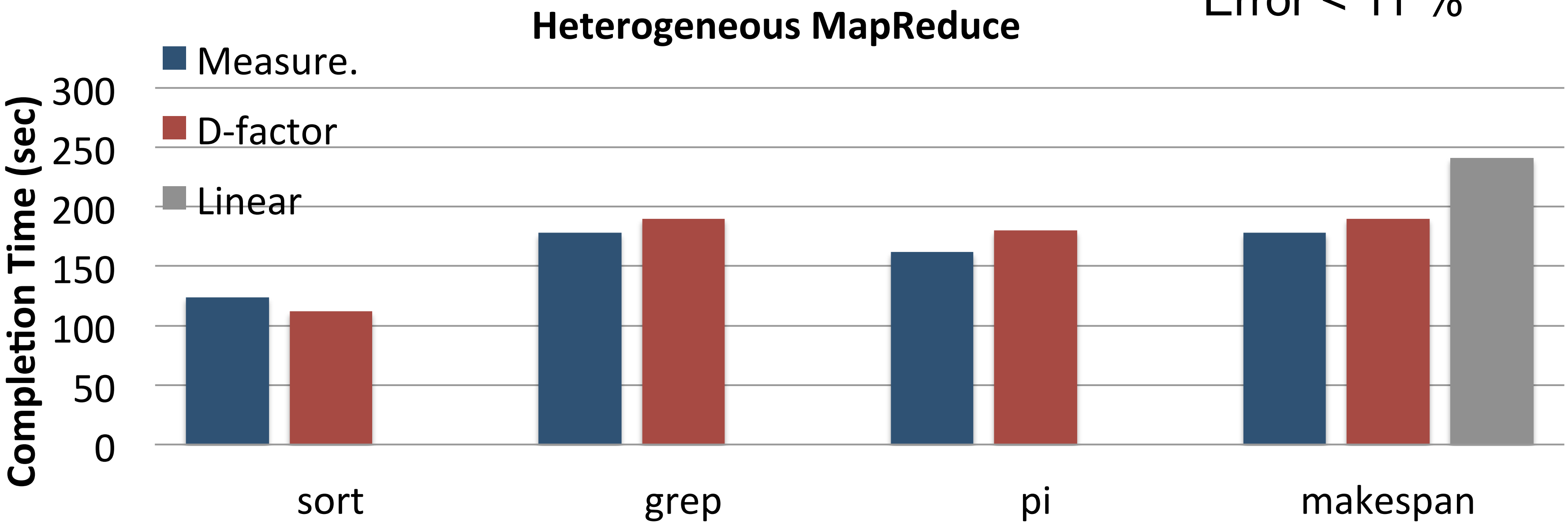


Identical workloads often shows the same phased behavior, which is hard to be explained with D-factor, which increases error rates as the number of instances increases.

Validation: MapReduce workloads

A 17 node Hadoop cluster results (1 master, 16 slaves)

Error < 11 %



Since heterogeneous workloads are more independent than identical workloads, error rates becomes smaller than identical workloads.

Summary

Performance Model:

We proposed a novel completion time model of jobs for shared service systems

- We modeled a job by a resource usage vector, called loading vector
- We showed that dilation factor of application slow-down can be modeled in a quadratic function of loading vectors.

Model Validation

We validated our proposed model with experiments using synthetic and realistic workloads.

How to use the Model in systems

We showed how to profile jobs and estimate the overall completion times of jobs in shared service systems

Future Work

Extending space-shared resources

(e.g., memory caches)

Developing a job scheduler with D-factor model

More validation with multi-core system

Questions?

Contact info

Youngjae Kim (PhD) / Seung-Hwan Lim (PhD)

kimy1@ornl.gov / lims1@ornl.gov
Oak Ridge National Laboratory